Fault Effects in FlexRay-Based Networks with Hybrid Topology

Mehdi Dehbashi, Vahid Lari, Seyed Ghassem Miremadi, Mohammad Shokrollah-Shirazi
Sharif University of Technology
{Lari, Dehbashi, Shirazi}@ce.sharif.edu
Miremadi@sharif.edu

Abstract

This paper investigates fault effects and error propagation in a FlexRay-based network with hybrid topology that includes a bus subnetwork and a star subnetwork. The investigation is based on about 43500 bit-flip fault injection inside different parts of the FlexRay communication controller. To do this, a FlexRay communication controller is modeled by Verilog HDL at the behavioral level. Then, this controller is exploited to setup a FlexRay-based network composed of eight nodes (four nodes in the bus subnetwork and four nodes in the star subnetwork). The faults are injected in a node of the bus subnetwork and a node of the star subnetwork of the hybrid network. Then, the faults resulting in the three kinds of errors, namely, content errors, syntax errors and boundary violation errors are characterized. The results of fault injection show that boundary violation errors and content errors are negligibly propagated to the star subnetwork. And syntax errors propagation is almost equal in the both bus and star subnetworks. Totally, the percentage of errors propagation in the bus subnetwork is more than the star subnetwork.

1. Introduction

Nowadays, Distributed embedded systems have significant position in modern industry because the distributed embedded control systems possess many advantages over traditional centralized ones, such as improved performance, optimized resource utilization, reduced cabling, enhanced modularity, and fault tolerance [1]. In a distributed system, each node consists of three parts [2]: 1) I/O part, 2) host part, and 3) communication controller. Among these three parts, the communication controller has a key role in the system operation.

In general, communication activities can be triggered either dynamically, in response to an event (event-triggered), or statically, at predetermined moments in time (time-triggered). Examples of time-triggered protocols are the SAFEbus [3], SPIDER [4], and Time-Triggered Protocol (TTP) [5]. The main drawback of the time-triggered protocols is their lack of flexibility [6]. Examples of event-triggered protocols are the Byteflight [7] introduced by BMW Company for automotive applications, CAN [8] and LonWorks [9]. The main drawback of the event-triggered protocols is their lack of predictability. A large consortium of automotive manufacturers and suppliers has proposed a hybrid type of protocol, namely, the FlexRay communication protocol [10] . The FlexRay allows the sharing of the bus among event-triggered and time-triggered messages, thus offering the advantages of both protocols. It is reported that the FlexRay will very likely become the de-facto standard for in-vehicle communications [6] [11]. The FlexRay defines a communication cycle (bus cycle) as the combination of a time-triggered (or static) window, an event-triggered (or dynamic) window, a symbol window and a network idle time (NIT) window. The FlexRay network is very flexible with regard to topology and transmission support redundancy [11]. It can be configured as a bus, a star or hybrid combinations of bus and star topologies.

The importance of safety in critical distributed applications signals to pay specific attention to the reliability of communication protocols. One way to assess the reliability of communication protocols is by fault injection. In [12], a simulation-based fault injection has been used for the assessment of message missings in the CAN network with bus topology. Effects of masquerade failures have been investigated using a simulation-based fault injection in the CAN network with bus topology [13]. Evaluation of TTP/C communication controller by heavy-ion fault injection (hardware-based fault injection) has been performed in

0-7695-3102-4/08 $25.00 © 2008 IEEE
DOI 10.1109/ARES.2008.161

[14]. The purpose of the experiments in that paper was to validate the fail silence property of the TTP/C by injecting faults in a single node. The relationship between the number of nodes in a cluster and the slightly-off-specification (SOS) failures has been assessed using heavy-ion fault injection [15]. In [16], the TTP/C protocol with bus and star topologies has been investigated using SWIFI fault injection. Here, the effects of the SOS failures in the bus and star topologies with respect to the start of frame transmission have been studied. In [17] [18] [19], a generic tool was developed for monitoring and diagnosis of a FlexRay-based system as well as for a CAN-based system. This tool has been used by the FlexRay consortium to perform extended fault injection for evaluating of the FlexRay communication protocol. One important limitation of this tool is that faults cannot be injected inside different parts of the FlexRay protocol.

This paper evaluates the fault effects in the FlexRay-based networks by injecting about 43500 bit-flip faults inside different parts of this protocol. To do this, a FlexRay communication controller is modeled by Verilog HDL at the behavioral level. This HDL model of the controller is exploited to setup the FlexRay-based network with hybrid topology. This network consists of two subnetworks: a bus subnetwork composed of four nodes and a star subnetwork composed of four nodes. To evaluate the faults effects in this network and vulnerability of these two subnetworks to the faults injection, the faults are injected into two separated nodes: one node in the bus subnetwork and another node in the star subnetwork. Then, the faults effects resulting in the three kinds of errors, namely, content errors, syntax errors and boundary violation errors are observed in each subnetwork. The dependencies of fault locations to these three kinds of errors are assessed in two mentioned subnetworks. Here, the sensitivity of each subnetwork to fault injections is evaluated. Also, the error propagation results in these two subnetworks are compared after fault injections.

This paper is organized in six sections. Section 2, introduces the FlexRay protocol, and section 3 presents error models found in this protocol. The experimental organization is given in section 4, and the results are presented in section 5. The last section concludes the work.

2. FlexRay protocol

A consortium of major automotive companies which includes BMW, Bosch, DaimlerChrysler, General Motors, Motorola, Philips, and Volkswagen, is currently developing the FlexRay protocol. The FlexRay network is very flexible with regard to topology and transmission support redundancy. It can be configured as a bus, a star or a multistar. It is not mandatory that each station possess neither replicated channels nor a bus guardian, even though this should be the case for critical functions such as steer-by-wire.

At the MAC level, FlexRay defines a communication cycle as the concatenation of a time-triggered (or static) window, an event triggered (or dynamic) window, a symbol window and a network idle time (NIT) window. The communication cycles are executed periodically. The time-triggered window uses a TDMA MAC mechanism; a station in FlexRay might possess several slots in the time-triggered window, but the size of all the slots is identical (Figure 1). In the event-triggered part of the communication cycle, the mechanism is Flexible TDMA (FTDMA): the time is divided into so-called minislots, each station possesses a given number of minislots (not necessarily consecutive), and it can start the transmission of a frame inside each of its own minislots. A minislot remains idle, if the station has nothing to transmit which actually induces a loss of bandwidth. The symbol window is a communication period in which a symbol can be transmitted on the network. The NIT window is a communication-free period that concludes each communication cycle.

The FlexRay frame consists of three parts: the header segment, the payload segment and trailer segment. The FlexRay header segment consists of 5 bytes. These bytes contain a reserved bit, payload preamble indicator, null frame indicator, sync frame indicator, startup frame indicator, frame ID, payload length, header CRC and cycle count.

The payload segment contains 0 to 254 bytes (0 to 127 two-byte words) of data. Because the payload length contains the number of two-byte words, the payload segment contains an even number of bytes. The FlexRay trailer segment contains a single field, a 24-bit CRC for the frame. The Frame CRC field contains a cyclic redundancy check code (CRC) computed over the header segment and the payload

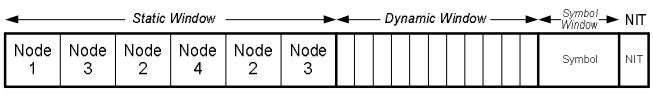

Figure 1. Communication cycle in FlexRay protocol

segment of the frame. The computation includes all fields in these segments.

3. Error models related to FlexRay

The FlexRay protocol has different mechanisms for detecting errors in the controller. At the end of each time slot, the FSP mechanism checks the presence of any error in that slot and informs the host about it. This protocol defines 3 main errors that can occur in each slot: syntax error, content error and boundary violation errors. The syntax error denotes the presence of a syntactic error in a time slot, the content error denotes the presence of an error in content of a received frame and boundary violation error denotes whether a boundary violation occurred at boundary of the corresponding slot.

4. Experimental Organization

This section discusses the basic characteristics of the experiment.

4.1. Experimental setup

For performing experiments, a FlexRay communication controller has been modeled by Verilog HDL at the behavioral level according to the FlexRay protocol specification [10] . This FlexRay controller has been tested according to the FlexRay protocol conformance test specification [20]. This HDL model of the controller has been exploited to setup a FlexRay-based network composed of eight nodes. The implemented controller has usual capabilities of the FlexRay protocol such as sending and receiving the static and dynamic frames and symbols. This controller according to the FlexRay protocol specification has six parts to perform its functions: controller host interface (CHI), protocol operation control (POC), clock synchronization process (CSP), frame and symbol process (FSP), media access control (MAC), coding and decoding (CODEC). In addition, instead of a real application, a data generator has been implemented to generate static frames with fixed length and dynamic frames with variable length at the start of the communication cycles.

The network topology in this experiment is a hybrid combination of bus and star topologies (hybrid topology). This topology with eight nodes is shown in figure 2. As depicted in this figure, this network includes two subnetworks: bus subnetwork and star subnetwork. In order to set up a network with hybrid topology, a model of central bus guardian (CBG) has been implemented at the behavioral level according to the FlexRay central bus guardian specification [21]. This CBG contains five branches that four nodes (nodes S1, S2, S3, and S4) are connected using point-to-point connections to four branches of the CBG. The fifth branch is connected to a bus topology that contains four nodes (nodes B1, B2, B3, and B4).

In this experiment, fault injection is done in two phases: 1) fault injection in one node of bus subnetwork (node B2), 2) fault injection in one node of star subnetwork (node S2). After each fault injection, error propagation observation is performed in both bus and star subnetworks (respectively in nodes B4 and S4). The faults are injected in five parts of the FlexRay communication controller, including CHI, POC, CSP, MAC and CODEC. As said in section 2.2, FSP part checks the correct timing and semantic correctness of received frames, and it applies further syntactical tests to received frames [10] . Thus, for the reason that the FSP part doesn't have any role in transmitting frames and error propagation to other nodes, there is no fault injection in the FSP part. The effects of fault injection are observed in FSP part of the FlexRay communication controller.

Central bus guardian (CBG). The CBG is an optional device that can be added to a channel of a FlexRay system in order to increase fault tolerance. The CBG guarantees that certain errors on one branch will not propagate to other branches by filtering functions. Examples of such filtering functions are: semantic filtering, content filtering (cycle count, and frame id filtering), Byzantine (SOS) filtering.

During normal operation the CBG enforces certain temporal aspects of the communication schedule. It does this via the use of several sub-states, each of which enforces different characteristics of the communication. During the static window the CBG operates in the strict sub-state. The CBG enforces a strict schedule within the static window by allowing only one slot/branch combination to send a frame. All other branches are blocked for transmission. The CBG disables transmission, if a frame is sent outside its timeslot. In the dynamic window the CBG operates in the dynamic sub-state (race arbitration). The first node/branch beginning to send in this segment is allowed to transmit. The protection always ends at the end of the frame. A node sending too long will be cut off when exceeding the maximum allowed frame length or the end of the dynamic window. During other portions of the cycle it operates in the idle sub-

state. In this sub-state the CBG disables all communication – no data is forwarded [21].

4.2. Fault injection tool

The SINJECT fault injection tool [22] is used for injecting fault at the behavioral level in nodes, collecting the results, and analyzing them. A fault injection process usually consists of three steps:

1- When the given workload is applied, the behavior of a fault-free network is computed and stored.
2- During the second step, to consider faults effects, the given workload are applied again to the network, the fault is injected, and the behavior of the network is observed.
3- During the third step of the fault injection process, the faulty network behavior is compared with the behavior of the fault-free network, which is gathered at first step, and therefore the fault effects are specified and saved.

5. Experimental Results

In this experiment for investigating the error propagations in a FlexRay-based network with hybrid topology, faults are injected in two separate nodes in this network: the node B2 in the bus subnetwork and the node S2 in the star subnetwork. In each of these two nodes about 21786 bit-flip faults are injected into five different parts of their communication controller. These five parts include: CHI, CSP, MAC, POC, and CODEC. Each experiment lasts for three communication cycles, in cycle 1 the faults are injected and the effects of them are observed in cycle 1 through 3. The error propagation observation is done in two different nodes: the node B4 in the bus subnetwork and the node S4 in the star subnetwork. In each communication cycle, 12 slot IDs in static window and 12 slot IDs in dynamic window are allocated to different nodes.

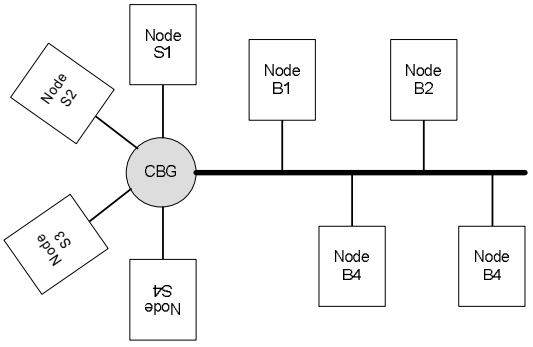

Figure 2. Experimental setup

The experimental results are investigated in 3 parts. In the first part the error propagation results are investigated as the result of fault injection in bus subnetwork. In the second part, the error propagation results are investigated as the result of fault injection in star subnetwork. Finally, in the third part, a comparison of error propagation results of previous parts is declared.

5.1. Error propagation after fault injection in bus subnetwork

In this part the faults are injected in different parts of a node in the bus subnetwork. After the fault injection in this node, the error propagation results are observed in another node in the bus subnetwork and a node in the star subnetwork. The errors are divided into three main classes. These three classes include syntax error, content error and boundary violation error.

Table 1 shows the errors that are observed in bus subnetwork. The CSP part is the most vulnerable to fault injection and the fault injections in this part lead to the most content errors, syntax errors and boundary violation errors. Also, in table 2 which shows the errors that are observed in star subnetwork, the CSP part is the most vulnerable part to fault injection. Fault injection in the POC part causes the least error propagation in the network.

Table 1. Error propagation in bus subnetwork (after fault injection in bus subnetwork)

FlexRay Parts	No. of Faults	Syntax Errors		Content Errors		Boundary Violation Errors	
		#	%	#	%	#	%
CODEC	5070	225	4.43	9	0.17	97	1.91
MAC	2196	174	7.92	119	5.41	104	4.73
CSP	8640	3059	35.4	1452	16.8	2658	30.76
POC	1680	2	0.11	0	0	0	0
CHI	4200	1304	31.04	485	11.54	400	9.52
All Parts	**21786**	**4764**	**21.86**	**2065**	**9.47**	**3259**	**14.95**

Table 2. Error propagation in star subnetwork (after fault injection in bus subnetwork)

FlexRay Parts	No. of Faults	Syntax Errors		Content Errors		Boundary Violation Errors	
		#	%	#	%	#	%
CODEC	5070	225	4.43	0	0	0	0
MAC	2196	195	8.87	0	0	0	0
CSP	8640	2955	34.2	59	0.68	35	0.4
POC	1680	2	0.11	0	0	0	0
CHI	4200	1391	33.11	19	0.45	2	0.04
All Parts	**21786**	**4768**	**21.88**	**78**	**0.35**	**37**	**0.16**

By comparing the results of table 1 and 2, it can be seen that the error propagation is reduced in the star subnetwork significantly. Most of the content and boundary violation errors occurring in the bus subnetwork are eliminated in the CBG. This is because of the fact that the CBG performs boundary protection and content filtering. This device disconnects the transmitter node/branch when it observes slot boundary termination or content error. Thus, the CBG prevent the propagation of the content and boundary errors from the bus subnetwork to the star subnetwork. The syntax errors propagation is almost equal in the both bus and star subnetworks. Totally, if a node becomes faulty in the bus subnetwork, error propagation in the bus subnetwork is more than the star subnetwork.

5.2. Error propagation after fault injection in star subnetwork

In this part the faults are injected in different parts of a node in the star subnetwork. Like the last part, the error propagation results are observed in another node in the bus subnetwork and a node in the star subnetwork.

Tables 3 and 4, respectively, show the error propagation in the bus subnetwork and the star subnetwork after fault injection in a node of the star subnetwork. In both of them, the CSP is the most vulnerable part to the fault injection among the different parts of the communication controller. The fault injections in this part lead to the most content errors, syntax errors and boundary violation errors. The POC is the least sensitive part to the fault injection and causes the least error propagation in the network. As these two tables show, like the last part, the error propagation in the star subnetwork is less than the error propagation in the bus subnetwork. It means, in spite of the faults are injected in the star subnetwork but the error propagation in the bus subnetwork is more than the star subnetwork. This is because of the fact that the faulty node can affect operation of other nodes in dynamic window while the CBG operates in the race-arbitration sub-state in this window. Thus in the bus subnetwork that the nodes are not controlled, the operation of faulty node can cause more errors in the bus subnetwork than star subnetwork in dynamic window. Totally, the results show that if a faulty node exists in the star subnetwork with hybrid topology, the error propagation in the bus subnetwork is more than the star subnetwork.

5.3. Comparison of error propagation in bus and star subnetworks

In the two past parts, the results of fault injections in bus and star subnetworks were investigated. According to these results, the CBG has an effective role in error propagation prevention between bus and star subnetworks. This device protects the star subnetwork nodes against the some errors propagation. As discussed in section 5.2, the star subnetwork is more fault-tolerant even against those faults that are injected in it. As in the bus subnetwork the bus is used as a common media by the nodes and transmissions are not controlled, the probability of error occurrence is higher than the star subnetwork. Thus, the bus subnetwork is more vulnerable against the fault injections. Totally, if a node becomes faulty in the hybrid topology (whether in the star subnetwork or in the bus subnetwork), the error propagation in nodes of the bus subnetwork is more than the star subnetwork. Also, the results show that entirely the CSP part of the FlexRay controller is the most vulnerable part and fault injection in this part causes the most error propagation in the network. The POC part is the least sensitive part to the fault injection.

Table 3. Error propagation in bus subnetwork (after fault injection in star subnetwork)

FlexRay Parts	No. of Faults	Syntax Errors		Content Errors		Boundary Violation Errors	
		#	%	#	%	#	%
CODEC	5070	210	4.14	3	0.06	0	0.00
MAC	2196	193	8.79	11	0.50	5	0.23
CSP	8640	2766	32.01	407	4.71	86	1.00
POC	1680	4	0.24	0	0.00	0	0.00
CHI	4200	1391	33.12	68	1.62	21	0.50
All Parts	**21786**	**4564**	**20.95**	**489**	**2.24**	**112**	**0.51**

Table 4. Error propagation in star subnetwork (after fault injection in star subnetwork)

FlexRay Parts	No. of Faults	Syntax Errors		Content Errors		Boundary Violation Errors	
		#	%	#	%	#	%
CODEC	5070	211	4.16	0	0.00	0	0.00
MAC	2196	200	9.11	0	0.00	0	0.00
CSP	8640	2766	32.01	84	0.97	24	0.28
POC	1680	4	0.24	0	0.00	0	0.00
CHI	4200	1403	33.40	13	0.31	0	0.00
All Parts	**21786**	**4584**	**21.04**	**97**	**0.45**	**24**	**0.11**

6. Conclusions

This paper investigated the error effects and error propagation in a FlexRay-based network with hybrid topology that includes a bus subnetwork and a star subnetwork. The investigation was based on about 43500 bit-flip fault injections inside five parts of the FlexRay protocol. To do this, a FlexRay communication controller was modeled by Verilog HDL at the behavioral level. A FlexRay-based network with hybrid topology composed of eight nodes was established using this controller. The results of fault injection showed that boundary violation errors and content errors are negligibly propagated to the star subnetwork. And syntax errors propagation is almost equal in the both bus and star subnetworks. Totally, the percentage of errors propagation in the bus subnetwork is more than the star subnetwork. Also the dependencies of fault locations to these three kinds of errors were assessed in two mentioned subnetworks. Here, the sensitivity of each subnetwork to fault injections was evaluated.

7. References

[1] J. Morris, D. Kroening, P. Koopman, "Fault Tolerance Tradeoffs in Moving from Decentralized to Dentralized Embedded Systems", *International Conference on Dependable Systems and Networks (DSN 2004)*, Italy, pp. 349-358, 2004.

[2] H. Kopetz, "A Comparison of CAN and TTP," *Vienna University of Technology*, Real-Time System Group, Research Report 23/1998.

[3] K. Hoyme, and K. Driscoll, "SAFEbus," *The IEEE Aerospace and Electronic Systems Magazine*, vol. 8, no. 3, pp. 34-39, 1992.

[4] P. S. Miner, "Analysis of the SPIDER Fault-Tolerance Protocols," *Proc. of the 5th NASA Langley Formal Methods Workshop*, 2000.

[5] H. Kopetz, and G. Bauer, "The Time-Triggered Architecture," *Proc. of the IEEE*, vol. 91, no. 1, pp. 112-126, 2003.

[6] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing Analysis of the FlexRay Communication Protocol," *Proc. of the 18th Euromicro Conference on Real-Time System*, pp. 203-216, July 2006.

[7] J. Berwanger, M. Peller, and R. Griessbach, "Byteflight-A New High Performance Data Bus System for Safety-Related Applications," BMW 2000, available in http://www.byteflight.de.

[8] R. Bosch GmbH, "CAN Specification," v2.0, 1991.

[9] Echelon, and LonWorks, "The LonTalk Protocol Specification," available in http://www.echelon.com.

[10] FlexRay Consortium, "FlexRay Communications System - Protocol Specification," v2.1 Revision A, December 2005.

[11] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in Automotive Communication Systems," *Proc. of the IEEE*, vol. 93, no. 6, June 2005.

[12] H. Salmani, and S. G.Miremadi, "Assessment of Message Missing Failures in CAN-based Systems," *Proc. of the Parallel and Distributed Computing and Networks*, pp. 387-392, 2005.

[13] H. Salmani, and S. G. Miremadi "Contribution of Controller Area Networks Controllers to Masquerade Failures," *Proc. of the 11th Pacific Rim International Symposium on Dependable Computing*, pp. 310- 316, 2005.

[14] H. Sivencrona, P. Johannessen, M. Persson, and J. Torin, "Heavy-ion Fault Injections in the Time-triggered Communication Protocol," *Proc. of the Latin American Symposium on Dependable Computing*, pp. 69-80, 2003.

[15] H. Sivencrona, M. Persson, and J. Torin, "Using Heavy-Ion Fault Injection to Evaluate Fault Tolerance with Respect to Cluster Size in a Time-Triggered Communication Systems," *Proc. of the IEEE International Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS-06)*, pp. 171-176, April 2003.

[16] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin, "Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Star Topology," *Proc. of the International Conference on Dependable Systems and Networks*, pp. 123-133 June 2003.

[17] R. Pallierer, M.Horauer, M. Zauner, A. Steininger, E. Armengaud, and F. Rothensteiner, "A Generic Tool for Systematic Tests in Embedded Automotive Communication Systems," *Proc. of the Embedded World Conference*, 2005.

[18] E. Armengaud, F. Rothensteiner, A. Steininger, and M. Horauer, "A Method for Bit Level Test and Diagnosis of Communication Services," *Proc. of the IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems*, 2005.

[19] E. Armengaud, A. Steininger, and M. Horauer, "An Efficient Test and Diagnosis Environment for Communication Controllers," *Proc. of the Austrochip Conference*, 2005.

[20] FlexRay Consortium, "FlexRay Communications System - Protocol Conformance Test Specification," v2.1, December 2005.

[21] FlexRay Consortium, "FlexRay Communications System - Preliminary Central Bus Guardian Specification," v2.0.9, December 2005.

[22] H. R. Zarandi, S. G. Miremadi, and A. Ejlali, "Dependability Analysis Using a Fault Injection Tool Based on Synthesizability of HDL Models," *Proc. of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 485-492, Boston, 2003.